Generating network-wide travel diaries using smartcard data

Oded Cats · Alex Vermeulen · Matej Cebecauer · Erik Jenelius · Yusak Susilo
AccessCard system in numbers

- Commuter train, metro, light rail, tram and bus
- 21 rail lines (470 km); 490 bus lines (9 100 km)
- Modal split

  - ~ 2 million tap-ins per day
  - Hybrid fare collection regime
    - Upon entering the station (underground, commuter train)
    - Inter-modes depends on transfer locations
    - Upon boarding the vehicle (bus)
    - Transition from on-board crew to ungated stations (tram and LRT)
**Routes Observed**

- **London - Bus**
  - Gordon et al. (2013)

- **Gatineau, Canada - Bus**
  - Trépanier et al. (2007)

- **Beijing - Bus**
  - Ma et al. (2013)

- **Stockholm Bus**

- **Netherlands - Bus & Tram**
  - Van Oort et al. (2015)
  - Luo et al. (2017)

- **South East Queensland, Australia - Bus, Train & Ferry**
  - Kieu et al. (2015)
  - Alsger et al. (2016)

- **Singapore - Bus**
  - Sun et al. (2014)
  - ... 

- **New York City - Metro**
  - Barry et al. (2002)
  - Barry et al. (2009)

- **Santiago, Chile - Bus & Metro**
  - Munizaga and Palma (2012)

- **Stockholm M + T**

- **London - Urban Rail**
  - Gordon et al. (2013)

- **Singapore - Metro**
  - Sun et al. (2012)

- **Shenzhen, China - Metro**
  - Zhao et al. (2016)
  - ...

**Routes Unobserved**

Source: Ding Luo
State of affairs

• Main mean of payment since 2010
• Anonymous or individual cards, with subscriptions
• Used for obtaining aggregate boarding numbers
• Project-based efforts to obtain passenger flows
• Privacy concerns hampering full-fledged utilization

• **FairAccess** project (2018-2019) the first to be granted access to data (GDPR compliant) for the sake of comparing (the fairness of) a fare scheme change

Combine destination, transfer and train inference algorithms in order to obtain a network-wide OD matrix estimation in the context of a mixed fare scheme system
Analysis process
Overall analysis flow

1. **Geospatial Data** (Stops, Lines, Locations)
2. **AFC Database** (Automated Fare Collection)
3. **AVL Data** (Automated Vehicle Locations)
4. **Data Processing Algorithm Development**
5. **Data Gaps / Problems**
6. **Data Analytics**
7. **Data Quality Assessment**
8. **VIA**: Vehicle Inferring Algorithm
   - **Vehicle Coupling**
9. **TOLIA**: TapOut Location Inferring Algorithm
   - **Tap Out Locations**
10. **TEA**: Time Estimation Algorithm
    - **Tap Out Times**
11. **Individual Diaries**
12. **Data Statistics**
13. **Slices/Subsets**
14. **OD-Matrices**
Inference algorithms: Alighting and transfer locations

[Trepanier et al. 2007]
Processing one user travel diary
Processing one user travel diary
Processing one user travel diary
Processing one user travel diary
Processing one user travel diary
Processing one user travel diary
Processing one user travel diary
Processing one user travel diary

Journey origin

11:06 M
11:17 R
11:12 M
15:34 B
Processing one user travel diary
Processing one user travel diary

TransitData
8 July 2019

4h gap

Journey origin

15:34

12:05

11:17

11:06

11:12
Processing one user travel diary

![Map showing a journey with different stops and times.]

- New journey origin
- Journey destination
- 11:06 M
- 11:17 R
- 11:12 M
- 15:34 B
- 12:05 R

TransitData
8 July 2019
Processing one user travel diary
Processing one user travel diary
Results
Hourly Tap-Ins

Hourly demand on average workday in mid-May

AM Peak
PM Peak
Passengers tap-ins and tap-outs (single legs)
Journeys origins and destinations
Interchange locations (downstream mode)
Mode migration flows

Transfer mode migration
Tap-in mode migration | 1st February 2017

Transfer mode migration
Combines tap-in mode migration with "metro - metro" transfer without tap-in | 1st February 2017
Fare zone migration flows
On-going and future activities

• ~84% of the journeys are inferred and included in the OD matrix (~87% of the trips)

• Passenger-vehicle assignment (metro, train) (Zhu et al. 2017)

• Flow distribution on links

• Method validation by
  – Contrasting with pass. counts
  – Applying in a system with ‘ground truth’ (e.g. Dutch)

• Price elasticity by user group

• Disruption detection
Case study
Modular Approach
Building Blocks

- Tap Out Location Estimation (TOLIA)
  Produces Likely Tap-Out Locations

- Time Estimation Algorithm (TEA)
  Adds Time Information for Tap-Out Estimations
  - Departure or Vehicle Inferring Algorithm (VIA)
    Adds Vehicle related data to Tap In/Tap Out Pairs

- Journey Algorithm (JA)
  Links trip elements into user journeys
Tap-Out Location Inferring Algorithm (TOLIA)

• Goal: match a tap-in its most likely tap-out using the information provided by the next tap in location
• A multi-modal implementation of Trepanier et al. (2007) and Munizaga et al. (2010)
• Key rules:
  – For in-Vehicle tap-ins the tap-out must lie along the line of the vehicle
  – Station tap-in means the tap-out must lie along a serviced route(s) for the given mode. This essentially means checking all stations having the same type that lie within a given distance of the next tap-in. This is because a transfer can be hidden “within” the tap-barrier zone.
  – Currently no Metro-Train connection without a transfer tap is considered. Future elements might implement this feature.
Time Estimation Algorithm (TEA) & Vehicle Inferring Algorithm (VIA)

• Goal: In order to predict if a tap in was made due to a transfer in an existing journey or to start a new journey a time needs to be estimated for the tap-out.
• for in-vehicle systems the time can easily be inferred given the tap-out location has been estimated using AVL data
• For station tap-ins the process is more complex and relies on the Vehicle Inferring Algorithm (VIA).
  – Direct: the most likely used vehicle is estimated based on the tap-in time, and then the arrival time of this vehicle on the estimated tap-out location is used as the tap-out time.
  – Transfer: a shortest path algorithm is used to generate intermediate transfer-stops and then for each of the legs the direct inference is used.
Journey Algorithm (JA)

• The Journey Algorithm is designed to detect journeys within a travel-diary (list of tapins & tapouts for a single CardKey). The Journey Algorithm takes the list of trips and then computes the time between a tap-out and the next tap in event. If this time is less than the transfer value input defined by the user then it infers a transfer and the two elements are assigned to a journey. If the time between the events is too big then a new journey is started and the old journey is completed. It does so for all trip elements meaning that the result of this step is a list of journeys made by the CardKey.

• This step also concludes the Origin Destination Searching as the journeys represent a list of Origins and Destinations when aggregated over all CardKeys.
Design Philosophy

• Modular
  – Data Availability
  – Solution Space Size
  – Features are separated for different uses

• Scalable
  – Lower Run times for testing
  – Validation for very small test set
  – Logical evaluation of a random element
  – Easy to explain the system to new users
Implementation

The framework is a collection of background process that provide backbone for the effective data analysis.

• **Input:** raw AFC and AVL data
• **Processes:** TOLIA, TEA, VIA, JA, Home base, ...
• **Output:** trips and journeys with their spatio-temporal information.
• **Additional processes for data analysis** – ODs, Crowding of vehicles, Disruption detection, patterns recognition, statistics, visualization, animation, ...

**Used technologies:**

• **Python**
• input/output can be any DB or files, but we use PostgreSQL DB with PostGIS extension

All used technologies are multiplatform and easily versatile (mac, windows, linux)
Implementation

Output in DB enables:

- multipatform access for many users and for different analysing tools or dashboards QGIS, ARCGIS, Tableau, Matlab, etc. at the same time.
- Additional background process for aggregation of data:
  - JA for journeys
  - Effective and fast cut/sampling of data (user, day, time of day, station, zone, etc ...)
  - Aggregate trips/journeys to ODs based on zones or stations
- Note just one month of trips is 8GB of data and it can easily overflow pc memory of the most personal computers if it is not pre-processed in DB.
Monthly Tap-Ins

Demand (tap-in) 2016/2017

- January: 55,000,000 (2016), 60,000,000 (2017)
- February: 55,000,000 (2016), 60,000,000 (2017)
- March: 65,000,000 (2016), 70,000,000 (2017)
- April: 60,000,000 (2016), 65,000,000 (2017)
- May: 60,000,000 (2016), 65,000,000 (2017)
- June: 50,000,000 (2016), 55,000,000 (2017)
- July: 40,000,000 (2016), 45,000,000 (2017)
- August: 50,000,000 (2016), 55,000,000 (2017)
- September: 60,000,000 (2016), 65,000,000 (2017)
- October: 70,000,000 (2016), 75,000,000 (2017)
- November: 60,000,000 (2016), 65,000,000 (2017)
- December: 50,000,000 (2016), 55,000,000 (2017)
Mode migration flows (II)

Transfer mode migration
combines tap-in mode migration with "metro - metro" transfer without tap-in | **morning peak** (06:00 - 09:00) 1st February 2017

Transfer mode migration
combines tap-in mode migration with "metro - metro" transfer without tap-in | **afternoon peak** (15:00 - 19:00) 1st February 2017